Environment/Architecture

Overview

Rishi 是易用的、鲁棒的时间序列预测工具。用户可以上传时间多时间序列文件、选择 feature，通过 Rishi 提供的模型或自定义的模型进行时间序列预测。Rishi 也将基于 Auto-ML，提供模型融合和调参的功能。

交互方式可以参考网站

Production Environment

- Software

  Rishi 预计通过 Container 进行 Web 应用的部署，目标的 Host OS 偏向于 Linux。同时每一个计算任务将运行在一个 Container 中，我们将可能借助 Kubernetes 进行 Container 的管理。

- Hardware

  Rishi 涉及对时间序列预测模型的训练，需要满足在计算量较大情况下的并发需求。初步设计中，当用户量较小时，我们的硬件应至少满足以下条件：

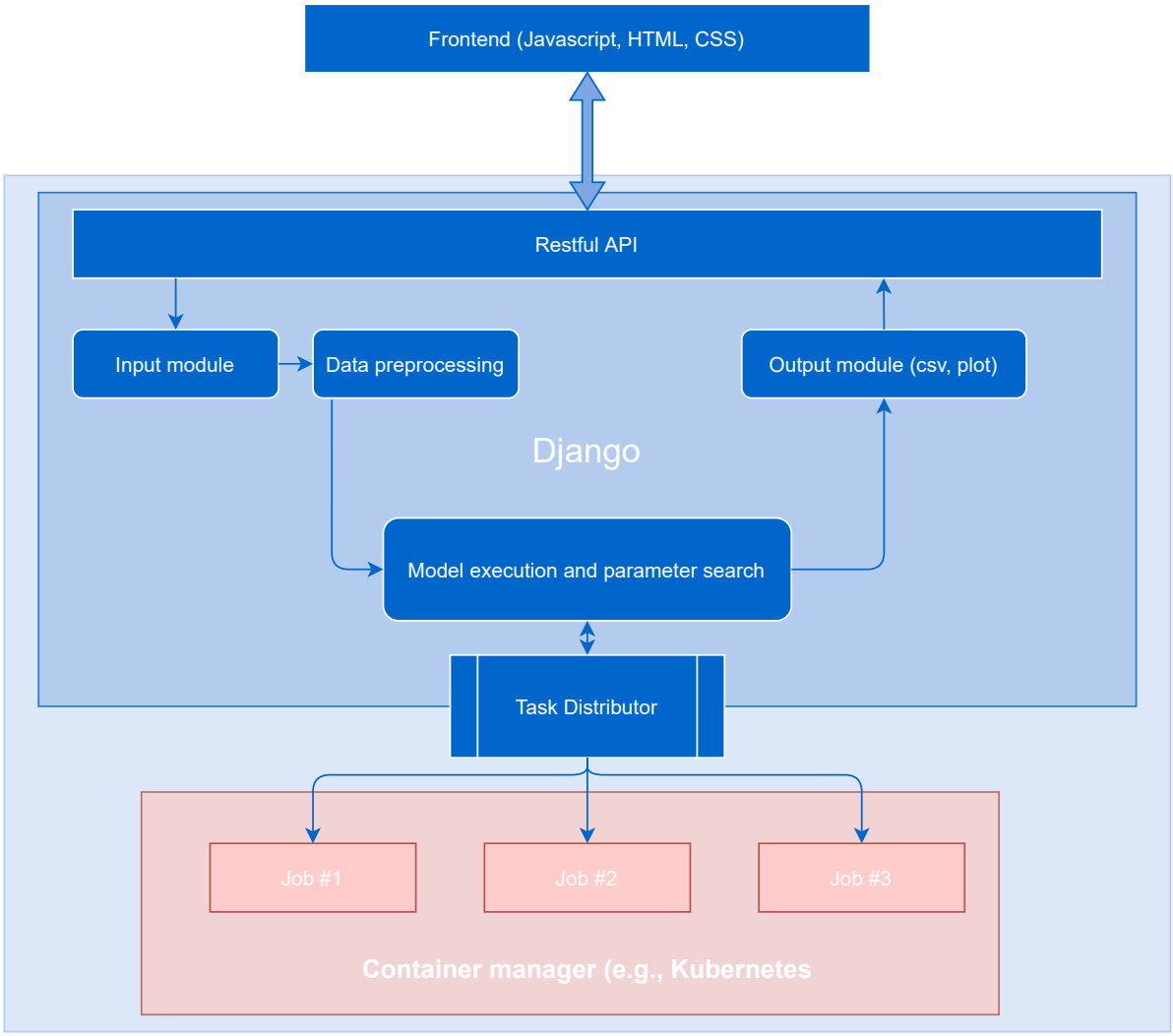
  - 16GB RAM
  - 8 Cores CPU
  - 1TB Storage: 由于需要存储用户任务训练后的模型和数据，因此会需要比较大的存储空间
  - Gigabyte Network: 满足大量数据的上传和下载。

  我们可能会借助云服务商来提供部署平台。

Architecture Design

分层设计

Rishi 将大致分为三层，其中一层为前端，将负责与用户进行交互，并向后端请求和返回数据。后端则将分为两部分，其中一部分是网络应用中 Django 为核心的一层，主要负责用户请求的处理和页面数据的返回。另外一部分是项目的核心部分，将负责计算任务的分发和执行，需要进行最佳参数的搜索和模型的训练。这一部分计算任务将运行在独立的 Container 中，并通过 Contrainer Manager (如 Kubernetes) 进行管理。



模块子系统设计

Rishi 将主要分成以下功能模块：

- User interface
- Input module
- Data pre-processing module
- Model execution and parameter search module
- Output module

由于 Rishi 预计通过 Django 以 Web App 的形式发布，因此在前后端的意义上，其后端主要需要有以下模块：

- Account, 提供用户登陆和管理账号信息的接口
- Project, 向用户提供建立项目，设定任务，得到结果等的接口
- Notification, 向用户展示通知

其它的模块则不需向前端提供 RESTFUL API 接口，而负责向后端提供其它内部接口。

时间序列预测模型

AdaptiveAvgModel

基础介绍

选用最优长度区间的均值进行预测

数学定义

$$\text{prediction} = \text{avg}(\text{max\_val}_n\{\sum_{i=0}^{n} ts[-(i+n):i]\})$$

超参数含义

1. round_non_negative_int_func(next_n_prediction_list : `list) -> list : func`
2. evaluation_function(pred : list, actual : list, model_name : `string) -> double : func`
   evaluation的函数
3. eval_len: `int`
   如数学定义中所述，定义了搜索范围

基本流程

$$\text{prediction} = \text{avg}(\text{max\_val}_n\{\sum_{i=0}^{n} ts[-(i+n):i]\})$$

在 `[1, eval_len + 1]` 的区间内遍历区间长度，找到最优长度区间，利用最优长度区间的均值进行预测

算法优劣分析

- 优势：算法较为简单
- 劣势：预测结果较为单一

常用场景

不适用于带有额外 feature 的情况

LinearFit

基础介绍

利用最小二乘法进行线性回归。

数学定义

$$w^*, b^* = \min_{w,b} ||y_i - (wx_i + b)||^2$$
$$\hat{y} = w^*x + b^* + \text{add\_std\_factor} \times \text{std}(y - (w^*x + b^*)) =$$

超参数含义

1. latest_n : `int` 用序列中latest_n项做预测
2. add_std_factor : `double` 控制训练中的标准差在预测结果中的比例

基本流程

用时间序列中 `latest_n` 项做线性回归，并加上线性回归的标准差

算法优劣分析

- 优势：算法较为简单，可以较好地刻画序列单调性特征
- 劣势：对于周期性、多次项的时间序列效果不好

常用场景

序列单调增长或减少，不适用于带有额外 feature 的情况

MaxNModel

基础介绍
用时间序列中`latest_n`项中最大值做预测

数学定义

$pred = \max(ts[-\text{latest\_n}:])$

超参数含义

1. lastest_n : `int` 用序列中lastest_n项做预测

基本流程

用时间序列中 `latest_n` 项中最大值做预测

算法优劣分析

- 优势: 算法较为简单
- 劣势: 预测效果差

常用场景
不适用于带有额外feature的情况

NewRandomArrivalModel

基础介绍
判断当前时间点是在上行段还是在下行段，根据事先选定的模拟上下行段的策略，进行预测

数学定义

指数、e指数、线性函数的数学形式

$$\lambda e^{-\lambda x}$$
$$y = a^x$$
$$y = wx$$

超参数含义

1. spike_detect_lag : `int`

   default: 12

   In spike detection algorithm, it describes the lag of the moving window

2. spike_detect_std_threshold : `int`

   default: 2

   In spike detection algorithm, the z-score at which the algorithm signals

3. spike_detect_influence : `double`

   default: 0.5

   In spike detection algorithm, the influence (between 0 and 1) of new signals on the mean and standard deviation

4. latest_n : `int`
   用latest_n项进行预测

5. rise_strategy : `enum{"exponential", "expectation", "linear, "auto"}`

   预测中上行段的策略

   1. "exponential": $\text{rise\_alpha}^{\text{rise\_step}}$
   2. "expectation": $\text{confidence} \times \text{avg\_spike\_height}$
   3. "linear": $\text{rise\_k} \times \text{rise\_step}$
   4. "auto": `pred_expectation if confidence < confidence_threshold else max(pred_exponential, pred_expectation)`

6. decline_strategy : `enum{"exponential", "expectation", "linear"}`

   预测中下行段的策略

    1. "exponential": $\text{decline\_alpha}^{\text{decline\_step}}$
    2. "expectation": $e^{\text{rise\_step}}$
    3. "linear": $\text{decline\_k} \times \text{decline\_step}$
7. confidence_threshold : `double`

   在上行段预测中的 auto 策略中，控制 confidence 阈值

8. height_limit : `enum{"average", "max_n"}`

   在上行段预测中，设置spike_limit的策略

    1. "average": `limit = avg_spike_height`
    2. "max_n": `limit = max(spike_height[-n:])`

基本流程

训练过程中，检测训练时间序列中的上行段和下行段，并计算以下参数用于预测

1. has_spike: 是否不是单调的
2. avg_rise_length: 上行段平均长度
3. avg_spike_height: 平均峰值
4. avg_decline_length: 下行段平均长度
5. avg_valley_height: 平均谷值
6. rise_k: $\frac{\text{avg\_rise\_length}}{\text{avg\_rise\_length}}$
7. rise_alpha: $\text{avg\_spike\_height}^{(1/\text{avg\_rise\_length})}$
8. spike_interval: 每一个峰值和谷值之间的间隔
9. expon_params: 用e指数函数对 spike_interval 进行回归，计算 e 指数的 lambda
10. last_spike_height: spike_height 中的最后一个
11. decline_alpha: $\text{last\_spike\_height}^{(1/\text{avg\_decline\_length})}$
12. decline_k: $\frac{(\text{avg\_spike\_height})}{\text{avg\_decline\_length}}$

预测阶段，判断当前时间点是在上行段还是在下行段，根据事先选定的模拟上下行段的策略，进行预测

预测中上行段的策略

1. "exponential": $\text{rise\_alpha}^{\text{rise\_step}}$
2. "expectation":$\text{confidence} \times \text{avg\_spike\_height}$
3. "linear": $\text{rise\_k} \times \text{rise\_step}$
4. "auto": `pred_expectation if confidence < confidence_threshold else max(pred_exponential, pred_expectation)`

预测中下行段的策略

1. "exponential": $\text{decline\_alpha}^{\text{decline\_step}}$
2. "expectation": $e^{\text{rise\_step}}$
3. "linear": $\text{decline\_k} \times \text{decline\_step}$

算法优劣分析

- 优势: 可以较好地刻画具有多个峰值的时间序列和周期性时间序列
- 劣势: 对于不符合指数、e 指数、线性概率分布函数的时间序列效果不好

常用场景

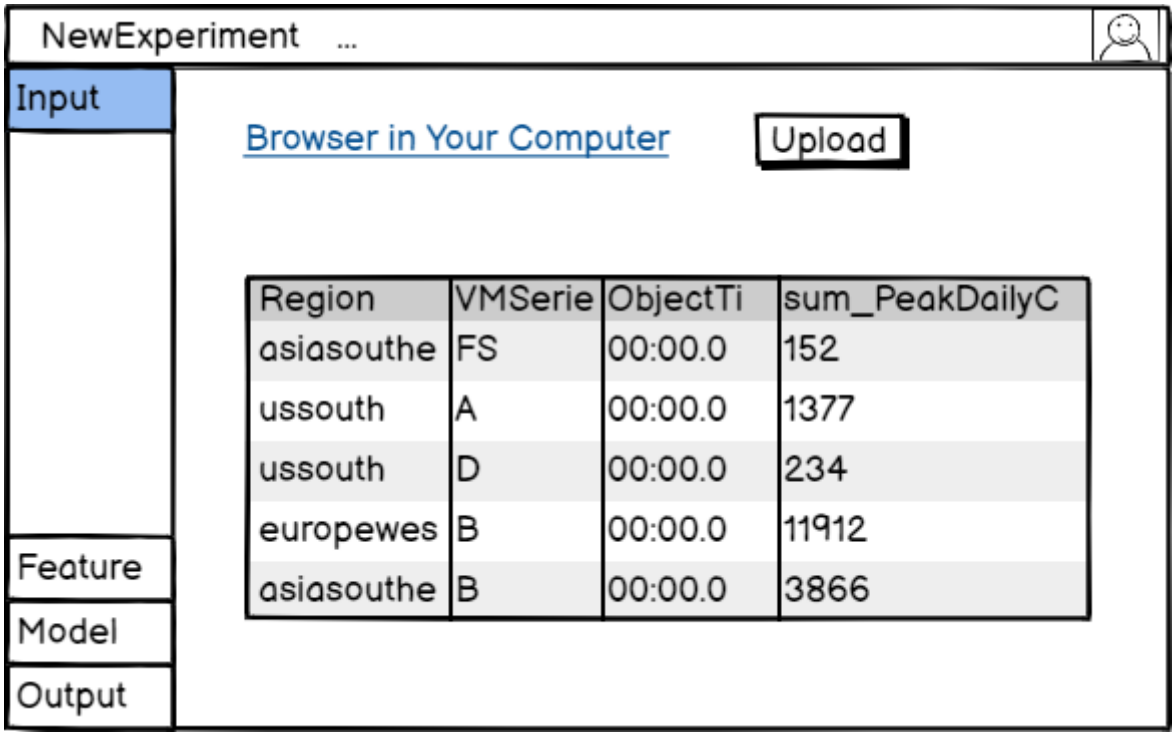
适用于有多个 spike 的情况，上行段和下行段符合指数、e 指数、线性概率分布函数

Interface/Interaction

用户接口

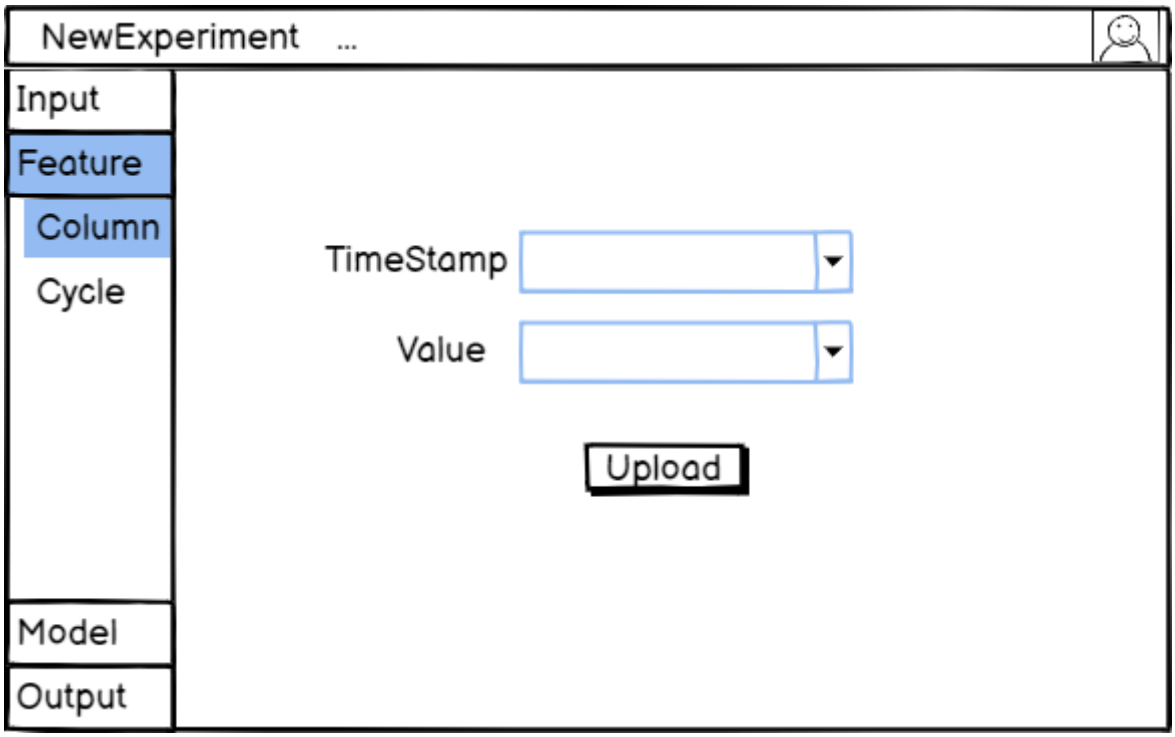
UI 基本按照处理步骤进行

Input



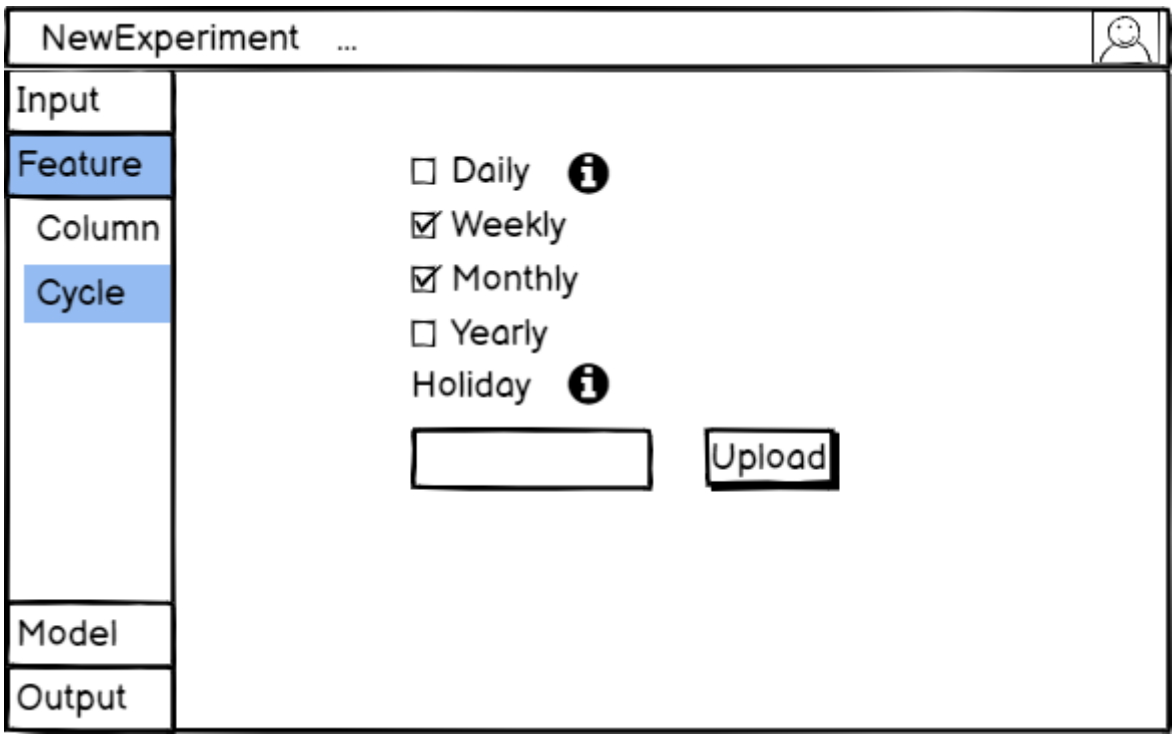
用户点击左上角 "Input" 选项，进入上传文件功能，限制用户上传的文件格式和大小，并且只上传一个文件。可以选择拖拽上传或者在文件资源管理器中选择文件两种方式，用户任选一种即可。文件上传成功或失败会在顶部弹出对应成功或失败的消息提示，如果上传成功，页面下方会显示上传的数据表格的前五行内容。数据上传成功后，进入下一步。

Select Column



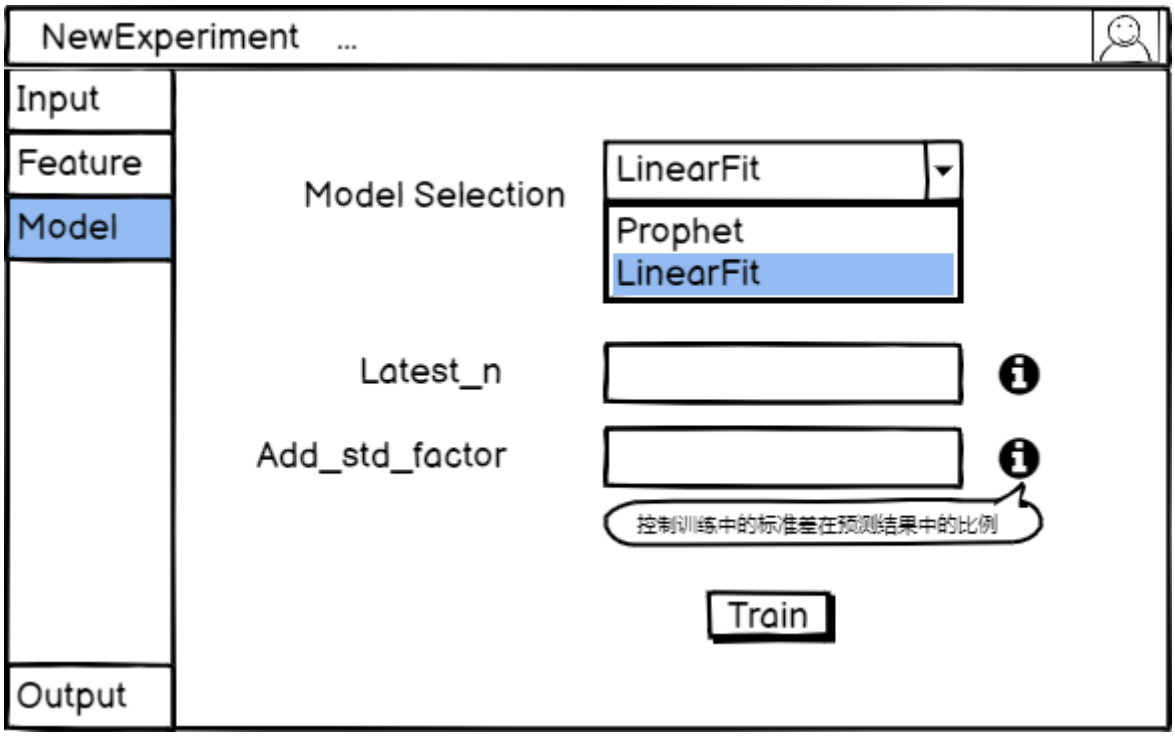
用户点击下拉菜单出现上传表格的列信息，需要用户选择表格中对应时间序列和值的两列的名称，点击Upload上传成功后进入下一步。[TODO]

Select Cycle



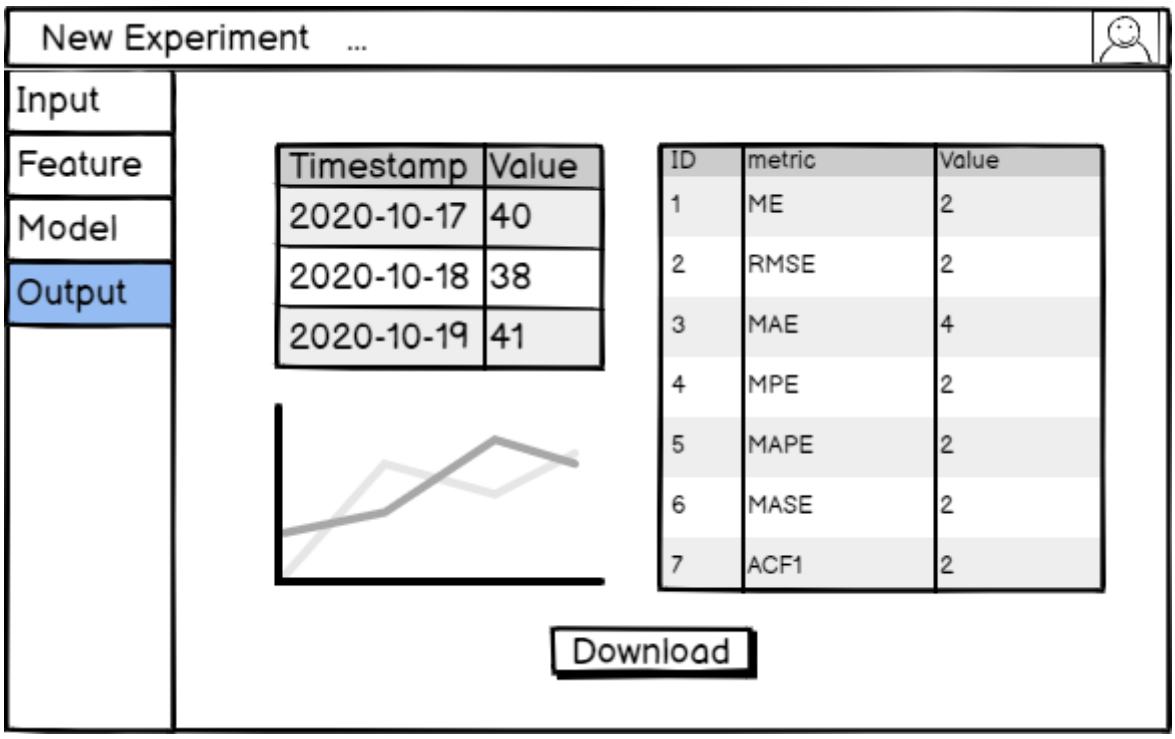
页面显示常见的时间序列预测中的特征值，包括但不限于关于不同时间单位的周期性以及假期等特征。用户根据实际情况勾选一个或多个周期单位，并按照提示的格式输入假期信息。如果用户对于选项存在疑问，可以通过将鼠标放置在 icon 上获得相应的帮助信息。用户输入和选择完成后，点击 upload 按钮提交。提交成功，进入下一步。[TODO]

Model



这个界面帮助用户完成模型以及相应参数的选择，同时提供信息帮助用户更好的进行决策。用户点击 Model Selection 对应的下拉菜单可以看到提供的多个不同的模型。用户选择一个模型之后，下方显示该模型对应的可调节的参数以及预训练的参数值。用户可以通过将鼠标放置在对应的 icon 上了解参数对应的含义以及推荐的调参方法。 在模型选择和参数调节结束之后，用户点击 "Train" 按钮开始训练。

Output



训练结束之后，以表格的形式输出预测结果。同时使用用户提供的时间序列、预测结果以及置信区间的相关信息绘制出曲线图。除此之外，包含

- ME: Mean Error
- RMSE: Root Mean Squared Error
- MAE: Mean Absolute Error
- MPE: Mean Percentage Error
- MAPE: Mean Absolute Percentage Error
- MASE: Mean Absolute Scaled Error
- ACF1: Autocorrelation of errors at lag 1

的表格展示模型的准确性。用户可以点击 "Download" 按钮，将结果保存下来。

程序接口

后端在分为 Architecture Design 中的 module 时，其提供的接口大致分为两种，由 Input 和 Output Module 为前端提供的接口，以及其它模块提供的内部接口。

RESTFUL API

本程序将使用前后端分离的方式，后端的 API 通过 RESTFUL API 提供。这一节将不会详细介绍每一个接口的参数，而将简略介绍每一个模块应当提供哪些接口，以及他们的作用。

account

- 用户登陆: 提供用户名和密码，负责将用户账户登入系统
- 用户注销: 注销用户
- 用户注册: 为新用户注册账号
- 用户 profile: 可以修改与账户关联的一些数据，例如用户头像，以及存储用户关于项目的默认设置等。

project

- 项目列表及创建: 在登陆的状态下，返回用户的所有项目列表。提供项目的表单，从而创建新的项目。
- 数据上传: 输入 csv 或其它特定格式的 dataset，每个项目应当可以拥有多个 datasets
- 数据预处理: 在已经输入 dataset 后，选定特定 dataset，返回一些信息（包括拥有多少 columns，数据格式，脏数据），提供选项给用户，目的是令用户选择时间以及目标值，以及选择数据预处理策略（剔除脏数据等）
- 项目设置: 修改 project 的 details，包括模型参数预设置，时间预测的周期，是否考虑季节性因素等
- Job Status: 由于模型的训练和 tuning 需要时间，需要通过此接口返回用户各个 job 的运行状态（Pending, Running, Success, Failed）
- Job Result: 向用户呈现任务的结果，通过图以及表格的方式。此接口需要返回数据点，对应图和表格的 render 应当在前端进行，从而更方便为用户提供可交互的图。

Notification

- 通知列表: 显示用户的所有通知，以及各通知的状态（已查看，未查看）

  通知会包括对用户项目中任务执行状态的通知（如任务已完成）

内部接口

此部分将介绍各个 module 需要向其它 module 提供的内容。

Data-preprocessing

- 分析
  - 输入：dataset
  - 输出：dataset 的分析结果，包括 columns，数据类型，脏数据
- 预处理
  - 输入: 用户的选项
  - 输出: 处理好的时间戳列和 y 值列

Model Execution and Parameter-search

此模块将是此项目的核心模块。其实现不局限于 python。容器需要编写 Dockerfile，对应的 job manager 可以通过其它语言实现，对应 search 的算法和模型选择也未知，可以是一个单独的应用，只需要提供以下的接口即可

- 增加任务
  - 输入: 模型，数据文件，以及此 job 的设置信息
  - 结果: 将此任务分配到某个容器中进行运行。此运行不确定什么时候开始及结束
  - 输出: 无
- 实时状态监控
  - 输出: 各个任务运行的状态信息，以及资源占用（资源占用会用于用户限额）
- 结果返回
  - 输入: 任务 id
  - 输出: 对应任务的结果

因此，此模块将通过此种方式和其它模块集成：

- 用户点击任务运行后，后端将此任务及 settings 推入此模块中
- 此模块负责任务的分配。并在过程中提供状态信息以及资源占用
- 后端检测到此任务执行完成后，会向此模块请求对应任务的结果，模块将结果返回

Static/Dynamic Data Design

数据结构及约束

根据思考调研，该项目实体型主要有用户(User)、数据(Data)、项目(Project)、作业(Job)、配置(Config)、模型（Model）几种。它们的数据结构及约束定义如下：

用户属性表(TE_USER)

中文名称	字段名称	数据类型	字长	主键	外键	唯一性	空值	备注
用户ID	USER_ID	INTEGER	4	是	否	是	否	自增
用户名	USER_NAME	CHAR	16	否	否	是	否	4-16位；由数字、字母、下划线组成
邮箱名	EMAIL	CHAR	32	否	否	是	否	需符合邮箱地址规范
密码	PASSWORD	CHAR	16	否	否	否	否	8-16位；由数字、字母、下划线组成 非对称加密算法加密后存储

用户上传数据表(TE_DATA)

中文名称	字段名称	数据类型	字长	主键	外键	唯一性	空值	备注
数据ID	DATA_ID	INTEGER	4	是	否	是	否	自增
数据名称	DATA_NAME	CHAR	32	否	否	否	否	由数字、字母、下划线组成
创建时间	CREATE_TIME	DATETIME	8	否	否	否	否	通过触发器设置
数据路径	DATA_PATH	CHAR	256	否	否	否	否	数据在服务器中存储路径
数据哈希	DATA_HASH	BINARY	32	否	否	否	否	通过触发器设置
用户ID	USER_ID	-	-	-	是	-	-	引用TE_USER.USER_ID

用户管理项目属性表(TE_PROJECT)

中文名称	字段名称	数据类型	字长	主键	外键	唯一性	空值	备注
项目ID	PROJECT_ID	INTEGER	4	是	否	是	否	自增
项目名	PROJECT_NAME	CHAR	32	否	否	否	否	由数字、字母、下划线组成
创建时间	CREATE_TIME	DATETIME	8	否	否	否	否	通过触发器设置
项目哈希	PROJECT_HASH	BINARY	32	否	否	否	否	通过触发器设置
用户ID	USER_ID	-	-	-	是	-	-	引用TE_USER.USER_ID

项目作业管理表(TE_JOB)

中文名称	字段名称	数据类型	字长	主键	外键	唯一性	空值	备注
作业ID	JOB_ID	INTEGER	4	是	否	是	否	自增
作业名称	JOB_NAME	CHAR	32	否	否	否	否	由数字、字母、下划线组成
创建时间	CREATE_TIME	DATETIME	8	否	否	否	否	通过触发器设置
修改时间	MODIFIED_TIME	DATETIME	8	否	否	否	否	通过触发器设置 为该作业最新配置提交时间
作业哈希	JOB_HASH	BINARY	32	否	否	否	否	通过触发器设置

作业配置管理表(TE_CONFIG)

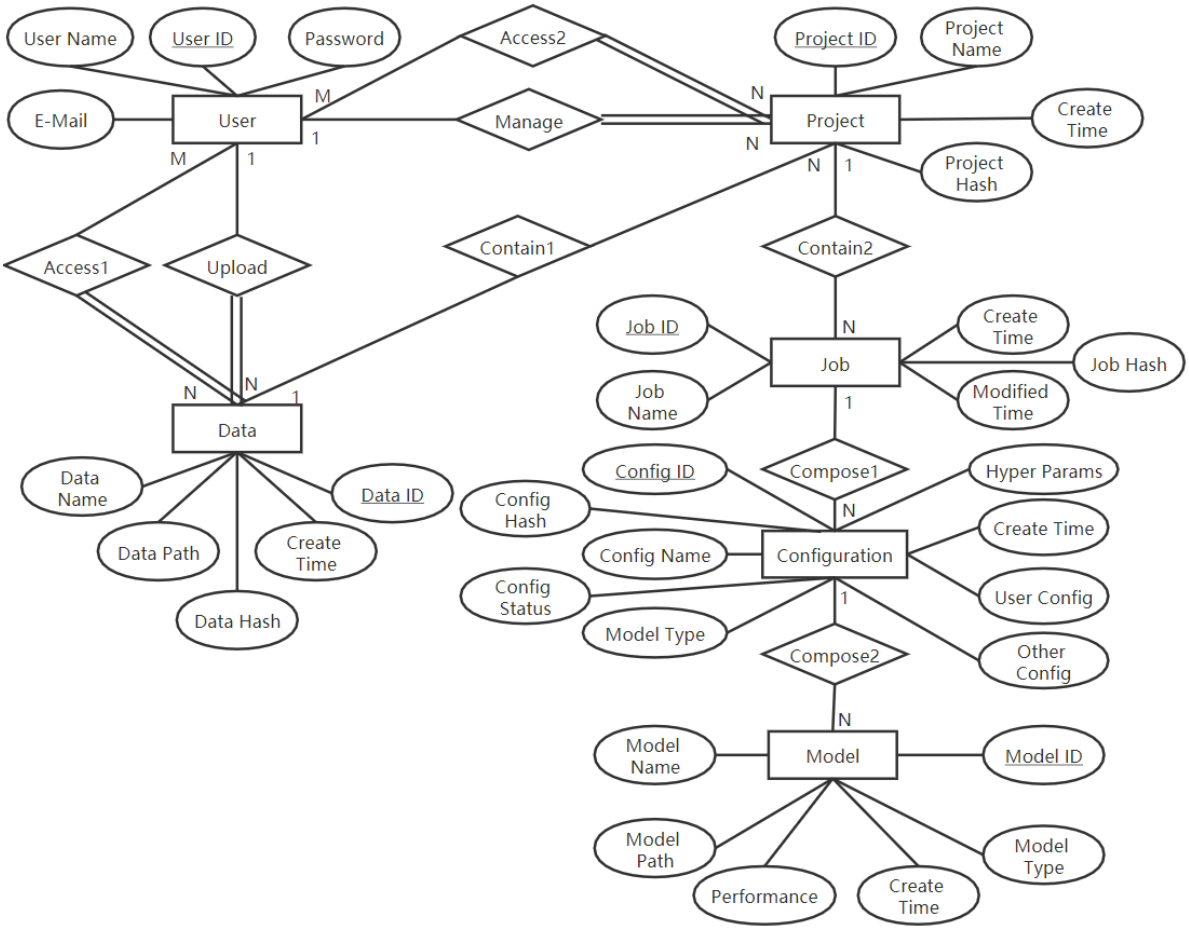
中文名称	字段名称	数据类型	字长	主键	外键	唯一性	空值	备注
配置ID	CONFIG_ID	INTEGER	4	是	否	是	否	
作业名称	CONFIG_NAME	CHAR	32	否	否	否	否	由数字、字母、下划线组成
模型类型	MODEL_TYPE	ENUM	2	否	是	否	否	-
超参数	HYPER_PARAMS	BLOB	216	否	否	否	否	-
自定配置	USER_CONFIG	BLOB	216	否	否	否	否	用户选定的数据列等
其他配置	OTHER_CONFIG	BLOB	216	否	否	否	否	-
创建时间	CREATE_TIME	DATETIME	8	否	否	否	否	通过触发器设置
配置哈希	CONFIG_HASH	BINARY	32	否	否	否	否	通过触发器设置
状态	CONFIG_STATUS	INTEGER	4	否	否	否	否	0 - 未提交 1 - 已提交在运行 2 - 运行结束未阅览 3 – 已阅览
作业ID	JOB_ID	-	-	-	是	-	-	引用 TE_JOB.JOB_ID

配置模型管理表(TE_MODEL)

中文名称	字段名称	数据类型	字长	主键	外键	唯一性	空值	备注
模型ID	MODEL_ID	INTEGER	4	是	否	是	否	自增
模型名称	MODEL_NAME	CHAR	32	否	否	否	否	由数字、字母、下划线组成
模型性能	MODEL_PERF	BLOB	216	否	否	否	否	精度、召回等 根据后续设置再调节表结构
模型路径	OTHER_CONFIG	CHAR	256	否	否	否	否	模型在服务器中的存储路径
创建时间	CREATE_TIME	DATETIME	8	否	否	否	否	通过触发器设置
模型哈希	MODEL_HASH	BINARY	32	否	否	否	否	通过触发器设置
配置ID	CONFIG_ID	-	-	-	是	-	-	引用TE_CONFIG.CONFIG_ID

概念数据库设计

根据需求分析和数据结构及约束，该项目的E-R图设计如下：



根据所设计的E-R图补全数据库中的联系型部分：

用户访问数据(TR_User_Access_Data)

键名	外键	基数
USER_ID	TE_USER.USER_ID	M
DATA_ID	TE_DATA.DATA_ID	N

用户访问项目(TR_User_Access_Project)

键名	外键	基数
USER_ID	TE_USER.USER_ID	M
PROJECT_ID	TE_PROJECT.PROJECT_ID	N

项目组成(TR_Project_Composition)

键名	外键	基数
PROJECT_ID	TE_PROJECT.PROJECT_ID	-
DATA_ID	TE_DATA.DATA_ID	-
JOB_ID	TE_JOB.JOB_ID	-

数据库的选用

由于该项目处理的数据大都是基于关系的，且暂无超大规模存储数据需求。我们使用关系型数据库作为基础解决方案。

考虑到生态规模、社区活跃度、花销以及该项目处于DEMO阶段，我们使用MySQL作为该项目的数据库支撑。

数据库基本支撑环境管理

基础软件系统

项目管理维护人员应提供满足数据库系统运行条件的基础软件系统，包括操作系统和数据库系统软件等。

操作系统

CentOS 7

数据库系统软件

MySQL 8.0

Process/Threading

Rishi 由于是个 Web App，其在 Web Server 部分的并发将主要由 Django Framework 来负责。

而此项目的核心是提供时间序列预测任务的 Training 和 Inference，涉及到大量的计算任务，因此需要比较完整的并发支持，包括资源限制，负载平衡。我们预计将通过容器的方式提供此部分任务的运行环境。并通过 Kubernetes 进行管理。